ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

OAuth 2.0 프로토콜에서 E-mail을 이용한 사용자 권한 인증☆

The Extended Authentication Protocol using E-mail Authentication in OAuth 2.0 Protocol for Secure Granting of User Access

채 철 주[1] 최 광 남[1] 최 기 석[1] 예 용 희[1] 신 용 주[1*]
Cheol-Joo Chae Kwang-Nam Choi Kiseok Choi Yong-Hee Yae YounJu Shin

요 약

최근 다양한 웹 서비스와 어플리케이션들이 사용자에게 제공되고 있다. 이러한 서비스들은 인증된 사용자에 한해서 서비스를 제공하기 때문에 사용자는 매번 서비스 별로 인증을 수행해야 하는 불편함을 겪고 있다. 이러한 불편함을 해결하기 위해 3rd Party 어플리케이션이 웹 서비스에 대하여 제한된 접근 권한을 얻을 수 있게 해주는 OAuth(Open Authorization) 프로토콜이 등장하게 되었다. 이러한 OAuth 프로토콜은 사용자에게 편리하고 유연한 서비스를 제공한다. 그러나 OAuth 2.0 프로토콜에서는 재전송 공격, 피싱 공격, 위장 공격 취약점이 발생할 수 있다. 그러므로 본 논문에서는 OAuth 2.0 프로토콜의 보안성 향상을 위하여 E-mail인증을 통해 Resource Owner를 인증 후 Access Token 발급을 할 수 있는 방법을 제안한다.

☞ 주제어 : OAuth 프로토콜, 인가, 인증, 접근 토큰, 이메일 인증

ABSTRACT

Currently there are wide variety of web services and applications available for users. Such services restrict access to only authorized users, and therefore its users often need to go through the inconvenience of getting an authentication from each service every time. To resolve of such inconvenience, a third party application with OAuth(Open Authorization) protocol that can provide restricted access to different web services has appeared. OAuth protocol provides applicable and flexible services to its users, but is exposed to reply attack, phishing attack, impersonation attack. Therefore we propose method that after authentication Access Token can be issued by using the E-mail authentication. In proposed method, regular user authentication success rate is high when value is 5 minutes. However, in the case of the attacker, the probability which can be gotten certificated is not more than the user contrast 0.3% within 5 minutes.

☞ keyword : OAuth Protocol, Authorization, Authentication, Access Token, E-mail Authentication

1. 서 론

A wide variety of web applications are published and in developments with rapid advancements in the technology of computer and networks. Of these applications the social network service (SNS) applications stand out. However, because of doing the basic authentication using ID/Password with the base, the user authentication has the structure of being vulnerable to the security. Therefore, the separate authentication method was developed for each web application and the user was authenticated in order to overcome this security vulnerability. This web application authentication method, for example, there is the Google AuthSub, AOL OpenAuth, Yahoo BBAuth, Amazon web service API, and etc[1-3]. Because of being different, the user required the method in which there was no separate authentication process, this authentication method can use for the application if it authenticated for each application.

Users wanted to share various web contents on their SNS page, and the OAuth protocol appeared in response to such users' needs[4]. OAuth protocol is a protocol that grants users access to multiple different services through just one time authentication between the third party application and the user [5-6]. Such protocol provides applicability and

[1] Dept. of R&D System Development, Korea Institute of Science and Technology Information, Daejeon, 305-806, Korea.
* Corresponding author (yjshin@kisti.re.kr)
[Received 21 October 2014, Reviewed 27 October 2014, Accepted 1 November 2014]
☆ This research was supported by the Sharing and Diffusion of National R&D Outcome funded by the Korea Institute of Science and Technology Information(KISTI).
☆ A preliminary version of this paper was presented at APIC-IST 2014 and was selected as an outstanding paper.

extensibility, but is vulnerable to attacks such as the replay attack, phishing attack, and impersonation attack. Such vulnerabilities can stir up some serious security issues because these security flaws can break in access to multiple services with only a single attack. For protection against such security vulnerabilities this study proposes a user authentication method that utilizes e-mail during the authentication between the third party application and the user.

The study is structured as followed. In Section 2, discusses the OAuth 2.0 protocol and its vulnerabilities. In Section 3, proposes the extended OAuth 2.0 protocol with e-mail authentication that protects against the security threats investigated in section 2. In Section 4, explains how the extended protocol protects against known security vulnerabilities and also tests out the performance of the extended protocol. In Section 5, discusses the conclusions.

2. Related Works

OAuth can not expose the user ID/PW to 3rd Party application and authenticate and the service provider can give the authority about authenticated API and accordingly provide API. OAuth announced the OAuth 1.0a which announced the OAuth 1.0 in 2007 and modifies the security problem of the OAuth 1.0 on June, 2008 and was registered as the IETF standard in 2010. Thereafter, the OAuth 2.0 draft was announced in 2012 and the OAuth 2.0 was registered as the standard on October, 2012.

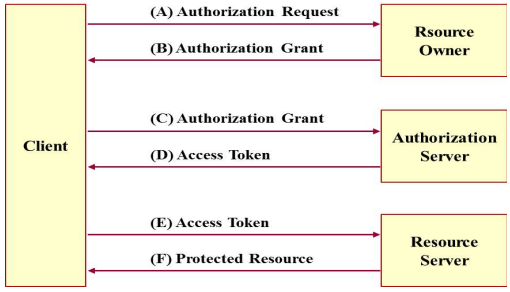
2.1 OAuth 2.0 Protocol

OAuth 2.0 protocol is an authentication protocol that grants user access to restricted services through onetime authentication using a third party application [7]. Currently OAuth 2.0 protocol enables comprehensive authentication to different applications of different service providers, and is in process of replacing the OAuth 1.0 protocol as a standard authentication method [8]. OAuth 2.0 protocol is structured as followed Table 1.

(표 1) OAuth 2.0 프로토콜 구성 요소와 역할
(Table 1) OAuth 2.0 Protocol Definitions

Symbol	Meaning
Resource Owner	An entity capable of granting access to a protected resource. When the resource owner is a person, it is referred to as an end-user.
Client	An application making protected resource requests on behalf of the resource owner and with its authorization.
Resource Server	The server hosting the protected resources, capable of accepting and responding to protected resource requests using access tokens.
Authorization Server	The server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization.

In OAuth 2.0 protocol the client asks the resource owner for authorization grants to the access to protected resources. The authorization grant is then delivered to the authorization server, and the access token is issued to the client after the user and client information are validated. Once the access token is issued the client can have access to the protected resource. Figure 1 shows the general process flow of authorization in the OAuth 2.0 protocol [9].



(그림 1) OAuth 2.0 프로토콜의 권한 인증 흐름
(Figure 1) Authorization flow of the OAuth 2.0 protocol

(A) A client requests the resource owner for authorization grant. The authorization grant can be requested directly or via the authorization server.

(B) The client receives the certificate containing the authorization grant of the resource owner.

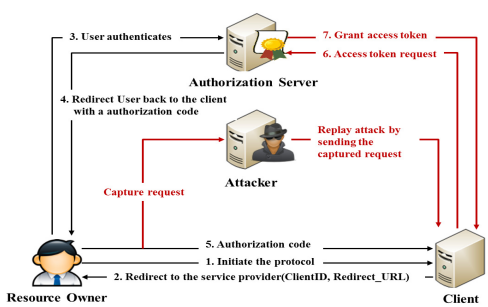
(C) The client uses the received authorization grant to ask an authorization server for an access token.
(D) The authorization Server authenticates the client and validates the authorization grant to issue the access token.
(D) The client uses the access token to ask a resource server for the access to protected resource.
(F) Resource Server returns the protected resource to the client if the access token is valid.

2.2 Analysis of security vulnerabilities in the OAuth 2.0 protocol

The common security vulnerabilities that can arise within the OAuth 2.0 protocol can be categorized into replay attack, phishing attack, and impersonation attack [10-13]. The followings show the analysis of each categorized attack.

2.2.1 Reply Attack

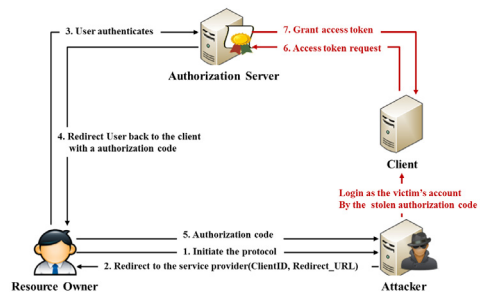
In OAuth 2.0 protocol, the authorization code means the client's access grant to the resource owner. Therefore one authorization code should be permitted per service. Otherwise a replay attack using the authorization code can be possible. The attacker can capture the authorization code redirection between the resource owner and the client, and use the captured request to replay the client to log-in as the account of the resource owner. The attacker can obtain the information of resource owner using such replay attack, and achieve the authorization grant to the resource server. Figure 2 shows how the replay attack works in the OAuth 2.0 protocol.



(그림 2) OAuth 2.0 프로토콜에서의 재전송 공격
(Figure 2) The replay attack in OAuth 2.0 protocol

2.2.2 Phishing Attack

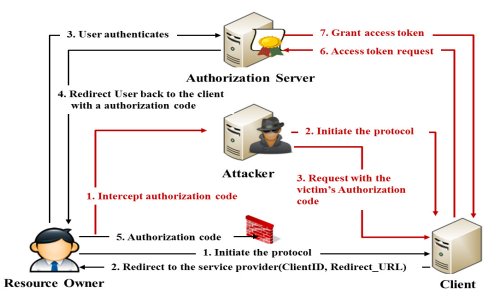
Phishing is very easy yet effective method for the attacker. The client must go through the authorization server in order to use the information of the resource owner. During this process the attacker can generate a malicious client to steal the ID and password of the resource owner. The attacker can fish for ID and password of the resource owner, and validly be granted the access to the resource server using the taken information. Figure 3 shows the phishing process in OAuth 2.0 protocol.



(그림 3) OAuth 2.0 프로토콜에서의 피싱 공격
(Figure 3) The phishing attack in OAuth 2.0 protocol

2.2.3 Impersonation Attack

The attacker can intercept the authorization code in attempt to impersonate the resource owner. The attacker first intercepts the authorization code by tapping, and blocks the original authorization code in order to maintain connection using the stolen authorization code. So the attacker can impersonate a fresh session with the stolen authorization code and have access to the resource server. Figure 4 shows the flow of impersonation attack in the OAuth 2.0 protocol.



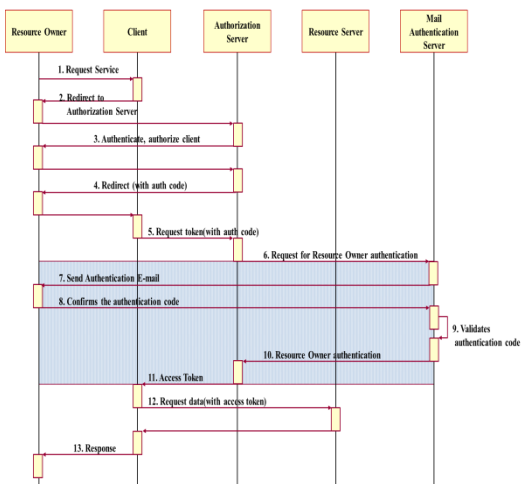
(그림 4) OAuth 2.0 프로토콜에서의 위장 공격
(Figure 4) The impersonation attack in OAuth 2.0 protocol

3. Extended OAuth 2.0 Protocol for Secure Granting of User Access

In this section, proposes a method to securely issue the access token, which grants access to the resource server, by authenticating the resource owner via e-mail in order to protect user from the security issues investigated in section 2. A list of symbols used in this paper is summarized in Table 2. Figure 5 shows the process of the extended protocol with e-mail authentication for more secure user access grant.

(표 2) 제안 논문 표기법
(Table 2) A list of symbols

Symbol	Meaning
user	Resource owner
U_{public}	User public key
$U_{private}$	User private key
MAS	E-mail authentication server
MAS_{ID}	MAS e-mail address
$USER_{ID}$	User e-mail address
M	Message
p	Prime number
g	Primitive generator
k	Random constant
MAC	Message authentication code
SK	Session key
T	Timestamp
h()	Hash function



(그림 5) OAuth 2.0 프로토콜에서의 위장 공격
(Figure 5) Extended OAuth 2.0 protocol with e-mail authentication for more secure user access grant

Step 1: Resource owner requests for the third party applications service, or the client.

Step 2: Client redirects the resource owner to the authorization server in order to authenticate the resource owner.

Step 3: Authorization server authenticates the client and resource owner.

Step 4: Authorization server redirects resource owner to the client with authorization code.

Step 5: Client uses authorization code to ask authorization server for issuing of the access token that grants access to the resource server.

Step 6: Authorization server asks MAS for authentication of the resource owner in order to verify that the client's request for access token was made by the resource owner.

Step 7: MAS sends an e-mail containing the authentication code to the resource owner.

Step 8: Resource owner confirms the authentication code in the inbox and sends back the confirmation to the MAS.

Step 9: MAS validates the confirmed authentication code e-mailed by the resource owner.

Step 10: MAS notifies the authorization server the authentication of the resource owner upon the confirmation of authentication code.

Step 11: Authorization server issues the access token to the client.

Step 12: Client uses the access token to access the protected resource on resource server.

For MAS to securely deliver the e-mail containing the authentication code to the resource owner in step 6, the ELGamal algorithm was used [14-15]. The MAS generates very large prime number p, and issue a public key, $y = g^a \bmod p$ that matches the private key, $a \in p^*$ of the resource owner signed up for the MAS. The MAS then goes through the following process in order to encode the authentication e-mail to be sent to the resource owner.

(1) Generate k smaller than p
(2) Generate a timestamp T_{MAS}

(3) Compute $SK_{MAS} = h(y^{k}_{U_{Public}}, MAS_{ID}, T_{AS})$

(4) Compute the codes $c_1 = g^k \bmod p$ and $c_2 = m \oplus sk_{MAS}$

(5) Compute $MAC_{MAS} = h(sk_{MAS}, USER_{ID}, m)$

(6) MAS sends the $\{MAS_{ID}, USER_{ID}, c_1, c_2, MAC_{MAS}, T_{MAS}\}$ to the resource owner

In step 7, the resource owner performs the following procedures in order to decode the $\{MAS_{ID}, USER_{ID}, c_1, c_2, MAC_{MAS}, T_{MAS}\}$ received from the MAS.

(1) Validate if the MAS_{ID} and $USER_{ID}$ valid e-mail addresses. If the address is not valid, the message is turned down.

(2) Validate T_{MAS} using $(T_{user} - T_{MAS}) \leq \triangle t$ where T_{user} is the generated timestamp upon the message receipt. If the validation fails, the e-mail message is turned down accusing for a replay attack.

(3) Compute the session key, $SK_{user} = h(c_{c_1}^{U_{private}}, MAS_{ID}, T_{MAS})$ using the private key of the Resource Owner.

(4) Restore message M by computing $M = c_2 \oplus SK_{user}$ and decoding the code, c_2.

(5) Compute and validate for equality of $MAC_{MAS} = h(sk_{user}, USER_{ID}, M)$. If the validation fails the resource owner turns down the e-mail message sent by the MAS, but if the validation succeeds the message M is saved.

(6) $MAC_{user} = h(SK_{user}, MAS_{ID}, USER_{ID}, M, T_{MAS})$ is computed and sent to the MAS.

(7) MAS computes the $MAC_{user} = h(sk_{user}, MAS_{ID}, USER_{ID}, M, T_{MAS})$ for equality with the value of received MAC_{user}. If the validation succeeds the resource owner is authenticated.

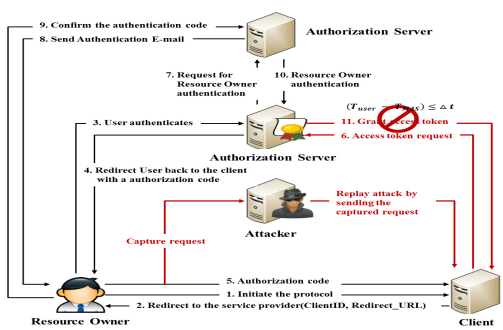
4. Evaluation of the proposed method

In this section, explains how the proposed method resolves security issues observed in section 2, and executes a performance evaluation of the proposed method.

4.1 Resolving of security issues of OAuth 2.0 using the proposed method

4.1.1 Preventing Reply Attack

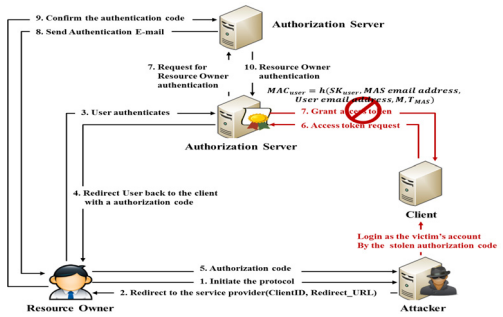
The replay attack captures the authorization code redirection request between the Resource Owner and Client and the attacker attempts the replay attack. However, in the proposed method, because of issuing the E-mail after authentication and Access Token, the attacker uses the replay attack and cannot approach the Resource Server. Because of verifying the validity of TMAS received through the verification process of the E-mail authentication procedure although the replay attack in which the attacker uses the E-mail in the authentication procedure is attempted, the replay attack of the attacker can be prevented.



(그림 6) OAuth 2.0에서의 재전송 공격 방지
(Figure 6) Preventing Reply Attack in OAuth 2.0

4.1.2 Preventing Phishing Attack

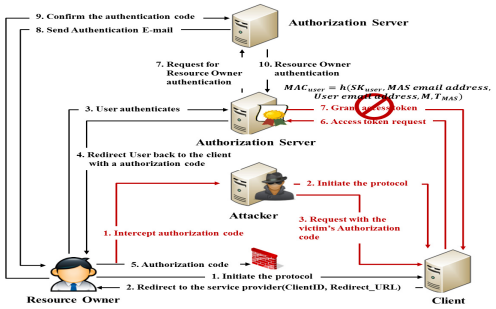
The phishing attacker creates a malicious client to phish for values entered by the resource owner and access to the resource. But such phishing can be prevented because the e-mail authentication goes through reciprocal authentication of MAC between the resource owner and the MAS.



(그림 7) OAuth 2.0에서의 피싱 공격 방지
(Figure 7) Preventing Phishing Attack in OAuth 2.0

4.1.3 Preventing Impersonation Attack

The impersonation attacker intercepts the authorization code by tapping, and blocks the original request to start up fresh and normal session using the intercepted authorization code. But use of e-mail authentication can prevent such impersonation attacks.



(그림 8) OAuth 2.0에서의 위장 공격 방지
(Figure 8) Preventing Impersonation Attack in OAuth 2.0

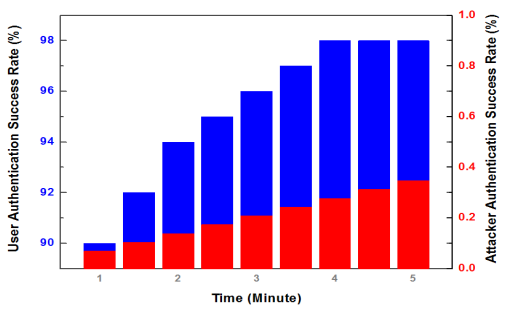
4.2 Performance evaluation

In order to test the effectiveness of the proposed method the timestamp value generated during e-mail authentication process is used. The performance evaluation is executed as followed. First the access token is issued through the e-mail authentication with the MAS when the resource owner makes a service request to the client. The MAS uses the timestamp to validate , so the access token to the resource server cannot be acquired unless the e-mail authentication is executed within the certain time.

For regular users, the authentication success rate goes higher as the timestamp value, which is the time constraint on the authentication process, becomes larger. For irregular attempts, the e-mail authentication will fail due to expiration of the timestamp since the resource owner would be not like to be aware of the e-mail sent by the MAS. So, the irregular attempt by the attacker would more likely to fail with smaller timestamp values. The followings are assumed for performance evaluation using the timestamp. The resource owner checks his or her e-mail at least once a day. The value does not exceed the maximum value of five minutes. Table 2 and figure 9 show the authentication success rate of regular user and attacker against varying timestamp values.

(표 3) 제안 방법의 인증 성공률
(Table 3) Authentication success rate

	Time (Minute)	E-mail comfirmation (count)	Attack possilility (count)	Success Rate (%)
User	1	1	1	90
	1.5	1	1	92
	2	1	1	94
	2.5	1	1	95
	3	1	1	96
	3.5	1	1	97
	4	1	1	98
	4.5	1	1	98
	5	1	1	98
Attacker	1	1	1,440	0.06
	1.5	1	960	0.1
	2	1	720	0.13
	2.5	1	576	0.17
	3	1	480	0.2
	3.5	1	411	0.24
	4	1	360	0.27
	4.5	1	320	0.3
	5	1	288	0.4



(그림 9) 정상 사용자와 공격자의 timestamp 값에 따른 인증 성공율
(Figure 9) Authentication success rate of regular user and attacker against varying timestamp values

Figure 9 show the regular user authentication success rate is high when value is 5 minutes. However, in the case of the attacker, the probability which can be gotten certificated is not more than the user contrast 0.3% within 5 minutes. Therefore, the safe user authorization is possible through the e-mail authentication. Therefore proposed extended protocol with e-mail authentication provide secure user access grant.

5. Conclusions

OAuth protocol is a protocol that grants users access to multiple different services through just one time authentication between the third party application and the user. Such protocol provides applicability and extensibility, but is vulnerable to attacks such as the replay attack, phishing attack, and impersonation attack. This study proposed a method to issue an access token after an e-mail authentication of the resource owner for enhanced security protection of OAuth 2.0 protocol. The proposed method can protect OAuth 2.0 protocol from known vulnerabilities such as the replay attack, phishing attack, and impersonation attack.

Because of overcoming the security vulnerability of the OAuth protocol, the proposed method provides the active service compared with the existing OAuth protocol. By preventing the security incident with the use of OAuth protocol using the proposed method afterward and applying the OAuth protocol to the moreover actively studied field of OpenID will be able to be activated.

참 고 문 헌 (Reference)

[1] Seon-Joo Kim, “An Efficient Access Control Mechanism for Application Software using the OAuth in the SaaS Cloud System”, Graduate School of PaiChai University, 2013.

[2] Jeong-Kyung Moon, “A Delegator for Authentication Management System using OAuth in Cloud Computing Environment”, Graduate School of Kongju National University, 2013.

[3] Myung Hyun Han, “Research on the extended OAuth protocol for real-name authentication”, Graduate of School of Information Technology Chung-Ang University, 2013.

[4] Meng-Yu Wu, Tsern-Huei Lee, “Design and Implementation of Cloud API Access Control Based on OAuth”, In Proc. Of TENCON Spring Conference, 2013. http://dx.doi.org/10.1109/TENCONSpring.2013.6584492

[5] http://en.wikipedia.org/wiki/OAuth

[6] Daeyoung Heo, Suntae Hwang, “OAuth based Proxy Delegation Service”, Journal of Internet Computing and Services, Vol. 13, No.6, 2012, pp. 55-62 http://dx.doi.org/10.7472/jksii.2012.13.6.55

[7] D. Hardt, “The OAuth 2.0 authorization framework,” Internet Engineering Task Force(IETF) RFC 6749, 2012.

[8] E. Hammer-Lahav, Ed, “The OAuth 1.0 Protocol”, Internet Engineering Task Force(IETF) RFC5849, 2010.

[9] M. Jones and D. Hardt, “OAuth 2.0 Authorization Framework: Bearer token usage”, Internet Engineering Task Force(IETF) RFC6750, 2012.

[10] Feng Yang, Sathiamoorthy Manoharan, “A security analysis of the OAuth protocol”, In Proc. Of Communications, Computers and Signal Processing (PACRIM), 2013. http://dx.doi.org/10.1109/PACRIM.2013.6625487

[11] T. Lodderstedt, M. McGloin, and P. Hunt, “OAuth 2.0 threat model and security considerations”, Internet Engineering Task Force(IETF) RFC6819, 2013.

[12] K. P. L. Francisco Corella, “Security analysis of double redirection protocols”, Pomcor, Tech. Rep., 2011.

[13] J. Richer, W. Mills, and H. Tschofenig, “OAuth 2.0 message authentication code (MAC) tokens”, draft-ietf-oauth-v2-http-mac-02, 2012.

[14] Won-Jin Lee, Kee-Won Kim , “Cryptanalysis and Improvement of an E-mail Exchange Protocol with Mutual Authentication”, Journal of KIIT, Vol 11, No. 10, 2013, pp. 61-68.

[15] Hae-Soon Ahn, Jongjung Woo, Ki-Dong Bu, ″Robust E-mail Exchange Protocol with Mutual Authentication″, Journal of KIIT, Vol. 10, No. 11, 2012, pp. 105-112.

◐ 저 자 소 개 ◑

채 철 주 (Cheol-Joo Chae)
2004년 한남대학교 컴퓨터공학과(공학사)
2006년 한남대학교 대학원 컴퓨터공학과(공학석사)
2009년 한남대학교 대학원 컴퓨터공학과(공학박사)
2009년 ~ 2013년 한국전자통신연구원 선임연구원
2013년 ~ 현 재 한국과학기술정보연구원 선임연구원
관심분야 : 정보보호, 네트워크 보안, 바이오 보안
E-mail : cjchae@kisti.re.kr

최 광 남 (Kwang-Nam Choi)
1992년 충남대학교 컴퓨터공학과(공학사)
1994년 충남대학교 대학원 컴퓨터공학과(공학석사)
1994년 ~ 현 재 한국과학기술정보연구원 책임연구원
관심분야 : 데이터베이스, 정보시스템, 네트워크
E-mail : knchoi@kisti.re.kr

최 기 석 (Kiseok Choi)
1998년 서울대학교 계산통계학과(공학사)
1997년 KAIST 대학원 정보 및 통신공학과(공학석사)
2013년 충남대학교 대학원 컴퓨터공학과과(공학박사)
1998년 ~ 현 재 한국과학기술정보연구원 책임연구원
관심분야 : 데이터베이스, 네트워크, 정보시스템
E-mail : choi@kisti.re.kr

예 용 희 (Yong-Hee Yae)
1978년 경북대학교 전자계산기공학과(공학사)
1991년 한양대학교 대학원 전자계산학과(공학석사)
1981년 ~ 현 재 한국과학기술정보연구원 책임연구원
관심분야 : 정보검색, 한글정보처리, 데이터베이스 구축
E-mail : yaeyh@kisti.re.kr

신 용 주 (YounJu Shin)
1986년 한남대학교 문헌정보학과(문헌정보학사)
2012년 한남대학교 대학원 문헌정보학과(문학정보학석사)
1986년 ~ 2006년 한국과학기술원 과학도서관사서
2006년 ~ 현 재 한국과학기술정보연구원 책임연구원
관심분야 : 데이터베이스, 정보검색, 정보서비스, 디지털 콘텐츠 구축
E-mail : yjshin@kisti.re.kr